Lux URP Essentials – URP 12 Preview

Since URP 12 introduces a huge amount of new features I will update Lux URP Essentials in several steps.

The first step includes most custom lighting functions implemented as .hlsl shaders. Custom lighting nodes for Shader Graph unfortunately do not work with deferred rendering and need Shader Graph to implement some new functionality. I asked for this but have no idea if it will come and when.

> Tested with Unity 2021.2.0b5. Other beta versions most likely will break the shaders.
>
> Screenspace shadows currently are broken in deferred and flicker. This can be fixed by enabling decals as well.

Currently supported shaders

The currently ported shaders are: Clear Coat (yeah still present as it offers options not available using the built in shaders), Cloth, Flat Shaded, Fuzzy Lighting, Skin, Hair, Lux Uber, Top down Projection, Terrain, Terrain Mesh, Toon, Transmission, Grass, Vegetation and Water.

Please open the URP12 Demo scene to get a better picture.

Custom Lighting

Custom lighting functions such as Cloth, Skin or Transmission can be used in forward and deferred rendering – however when it comes to deferred, materials using these functions will be rendered in forward rendering.

Due to the depth prepasses, decals and SSAO, they need an additional but rather cheap GBuffer pass as well :(*This however should only be true in case you have checked "Accurate G-buffer normals" in your deferred renderer. Otherwise you may try and comment out the GBuffer pass.*

Deferred Rendering

Just like in the built in render pipeline URP's deferred renderer does not allow us to use custom stencils – they are simply ignored. This means that stencil based fast outline features are not supported in deferred.

Deferred also does not offer any support for custom lighting functions which therefore will fallback to forward. See above.

Forward Rendering

Forward rendering now supports *Depth Priming* which is just what most people know as *depth prepass*. In case it is enabled or in case decals or SSAO are enabled (which also cause a depth prepass) *Alpha To Coverage* most likely will produce artifacts: We simply can not write more than a single value into the z-buffer. *So simply deactivate Alpha To Coverage on all materials in case you use Depth Priming, SSAO or decals.*

Shaders

Additional Surface Options

Most shaders contain two new custom surface options:

Enable Normal in Depth Normal Pass The built in Lit shader will output per pixel normals in the lit depth normal pass which makes it sample the normal map twice. In case you only have subtle normals or just need better performance you may uncheck this and the depth normal pass will skip the per pixel normals. *This affects SSAO and decals.*

Receive Decals HDRP offers this per material and so do Lux URP Essentials. So in case you do not want or need a material to receive decals just uncheck this option.

Uber

Uber's **parallax mapping** has been optimized for the new *Depth Priming* and will produce stable results even if alpha testing is enabled (unlike the built in lit shader. I filed a bug report about this…).

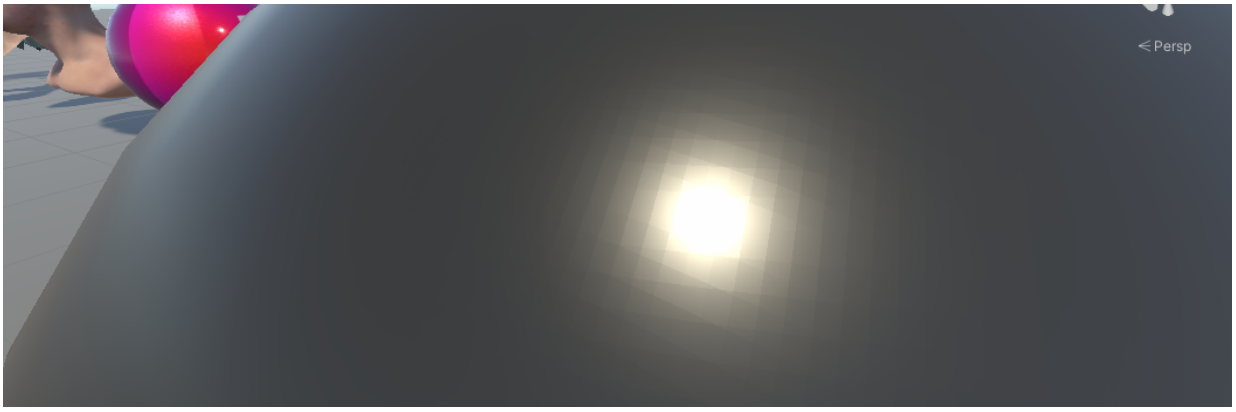
Uber now supports **best-fit normals** as invented by Anton Kaplanyan for Cryengine 3. These are useful in deferred rendering: The default quality of the GBuffer normals is far from being optimal (8bit per channel only) and will create a faceted look on sharp specular highlights.

So URP's deferred renderer offers *Accurate G-buffer Normals*. Activating this feature however means that all normals on all materials will be first encoded to OctQuadEncoded ones during the GBuffer pass and then decoded for each deferred lighting pass – which produces some overhead (and breaks the terrain add pass…).

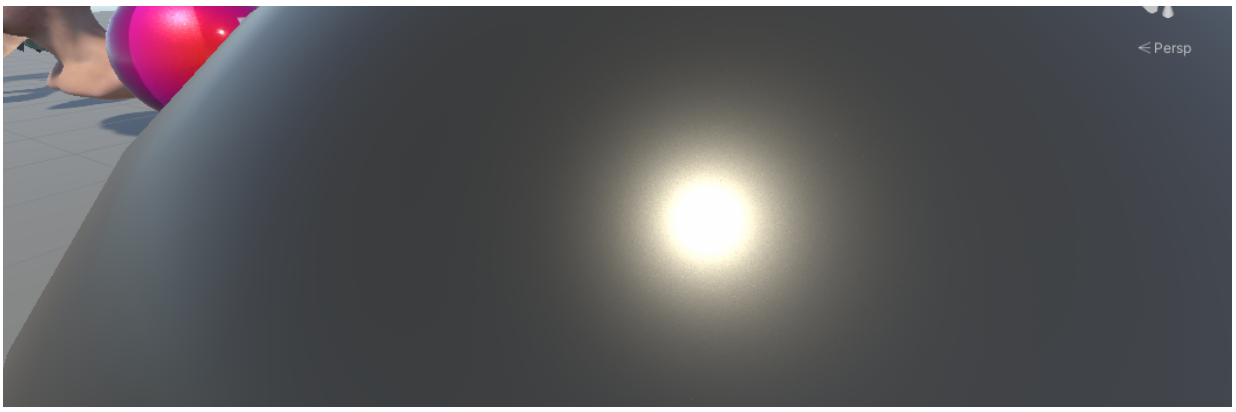
You might have a lot of materials where low quality normals would just be fine as you have a strong normal map or low smoothness so artifacts are barely visible.

Here *best-fit normals* come into play as you dedicatedly specify which material needs high quality normals and only use them there. Furthermore *best-fit normals* only need the encoding step, decoding is just the same as for regular written normals and virtually free.

So *best-fit normals* let you save a lot of computational power by adding the possibility to render high res normals only on selected objects/materials.



URP low quality GBuffer normals. Note the pixelated or faceted look of the highlight.



Uber using best-fit normals. Normals here still are not perfect and will show up some noise. But in most cases they look way better than the standard low quality normals. Using *Accurate Gbuffer normals* all across the board will outperform *best-fit normals* for sure but at higher costs on the GPU.

In case you opt for *Accurate G-buffer Normals best-fit normals* will automatically be optimized away.

Best-fit normals always only are applied in deferred rendering because in forward rendering we simply have nearly perfect normals anyway.

Toon Shader

Due to the new *Depth Priming* we can no longer use a single shader to draw the toon shaded surfaces plus the outline: As we only have one *Depth* or *DepthNormal* prepass the outline would simply be ignored. Thus we have to assign two materials to the renderer: First one for the toon shading and the second one to add the outline.

Grass

URP 12 allows us to use shaders supporting instancing within the terrain engine. So now you can place grass using the grass shader right within the terrain engine.

Grass supports *improved specular lighting* as I added the *Specular Mask* feature from ATG. Using this feature you should be able to add some nice reflections even if your grass model has heavily tweaked/smoothed normals.

It also comes with *Screen Space Normals* which are normals on single sided geometry not being tweaked based on *VFACE* but their orientation in screen space. So if enabled the shader will not flip or mirror the normals but just ensure that normals will always point towards the camera. This may soften lighting on some models using the grass shader (models whose normals are not fully softened).

Foliage

The foliage shader (which can also be used within the terrain engine, just like grass) offers various techniques to add some kind of transmission even to *deferred rendering*, which are: Standard, Simple, NormalVS, Transmission.

> **Standard** Well, this mode does not support any kind of transmission… Normals of back faces will be corrected using *VFACE* *Cheap*
>
> **Simple** Just like *SpeedTree* the shader will not flip the normals based on *VFACE*. So some faces will show up proper front face lighting, some will show up false back face lighting. This will create the *illusion* of transmission but also may add a lot of false specular highlights. *Cheapest.*
>
> **NormalVS** This will not add transmission lighting but may correct and smooth lighting compared to *Standard* and *Simple*. Normals here are not flipped or mirrored based on *VFACE* but will be tweaked in screen space to always point towards the camera. *A bit more expensive.*
>
> **Transmission** This mode will add proper transmission lighting for the most dominant directional light only. Additional spot or point lights are not supported. This mode will sampe shadows and cookies in the deferred GBuffer pass (which usually does not do this at all) and thus makes it: *Most expensive.*

Wrapped diffuse lighting is not supported in deferred.

Terrain

Here Unity seems to have to add some more love:

- The Lit terrain shader shipping with URP 12 does not support bakes shadow maps. Lux Essentials' one does.
- When it comes to *Real Time Global Illumination* the original shader does not support this as well. Lux Essentials' one does somehow.

Please note: It is all a bit unstable. So in case the additional pass does not render properly leaving black spots on the terrain, the distance terrain using the basemap shader does not show up or anything else strange happens, try to toggle the terrain on and off.

Water

Water is always rendered using forward.

New inputs:

> **Diffuse Normal Up** Lets you tweak the normal that is used to calculate diffuse lighting on the underwater fog. 0.0: Only the geometry normal will be taken into account. 1.0:

Only a fully upwards pointing normal will be taken into account. Anything in between will lerp both normals.

Add Foam from Normal Lets you strengthen the foam distribution based on the water normals to break up foam. Default was 4.0.

Shader Graph and custom lighting

The provided Shader Graph nodes for custom lighting should still work when it comes to forward lighting (not tested tho...) but will not support any new features such as cookies or light layers.

They do not support deferred lighting. Actually they may produce a somehow proper output for just a single directional light – which they may sample and output to emissive. Any additional light however will be fully ignored: Their data is simply not bound.

Here Unity will have to add some functionality like declaring a pass as “Forward only”. I filed a request and we will have to see what they will come up with.